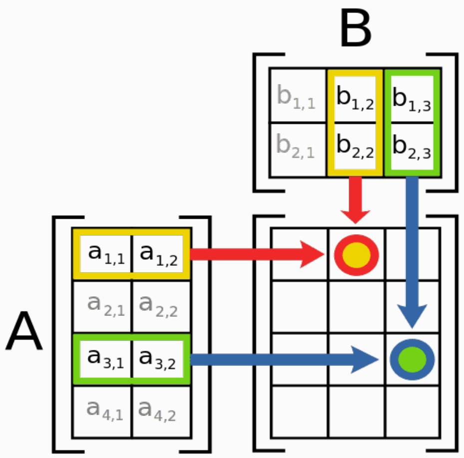
UNIVERSITY OF CAMBRIDGE

Topic: J. Xing et al.: Bolt: Bridging the Gap between Auto-tuners and Hardware-native Performance

Presenter: Woon

ML needs Auto-Tuning for Efficient Generation of Tensor Program



Machine Learning consists of many matrix multiplication operations.

Ansor

tvm

Ansor is an **Auto-tuner integrated into tvm** to generate efficient tensor program

Informally, a Tensor Program is just a composition of matrix multiplication and coordinatewise nonlinearities.

Effective implementation on hardware with high accuracy and efficiency

Tunable Parameters include: block size for tiling, loop order, unroll factor, and explicit memory movement to optimize cache locality and parallelism.

Auto-Tuning Rationale

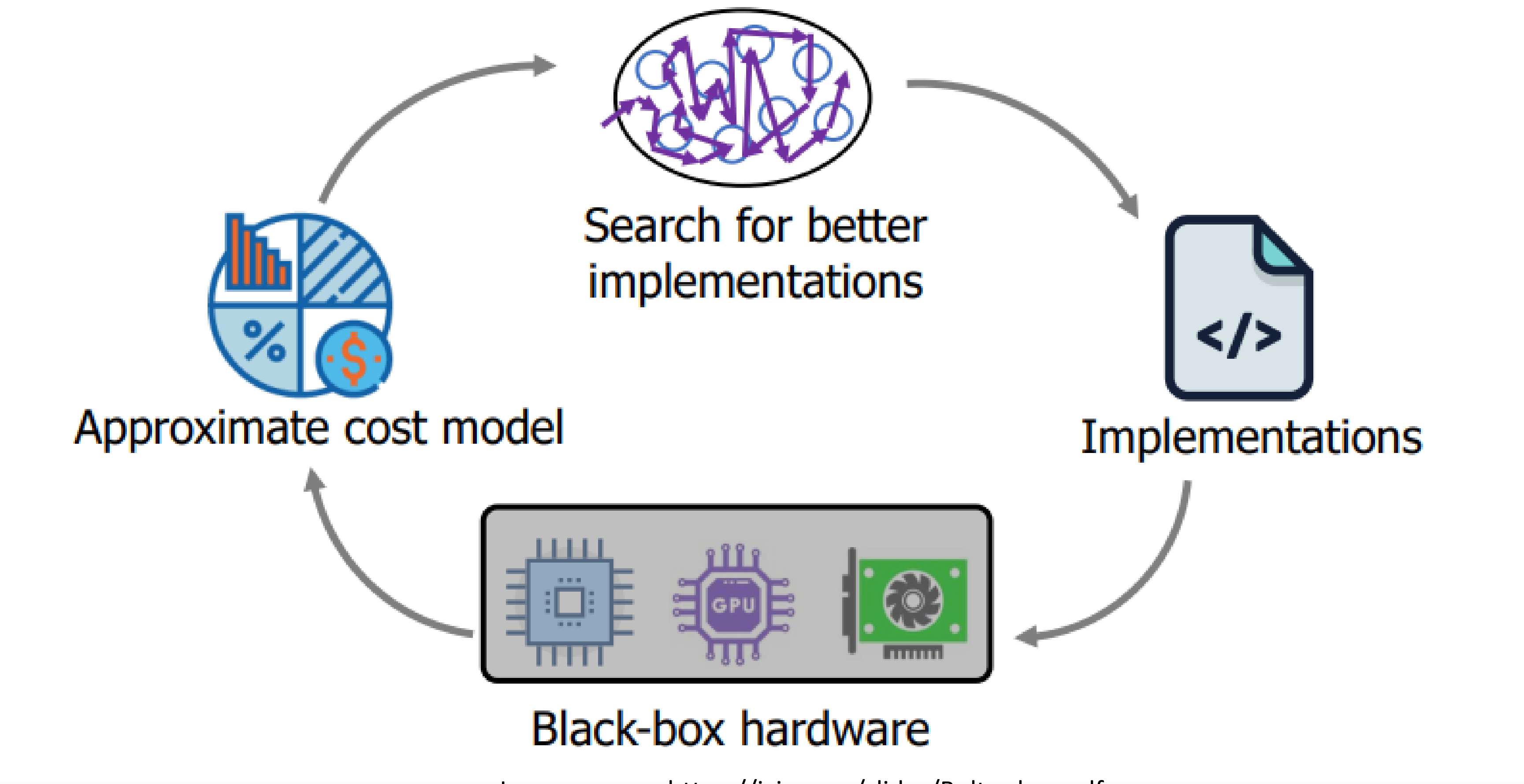
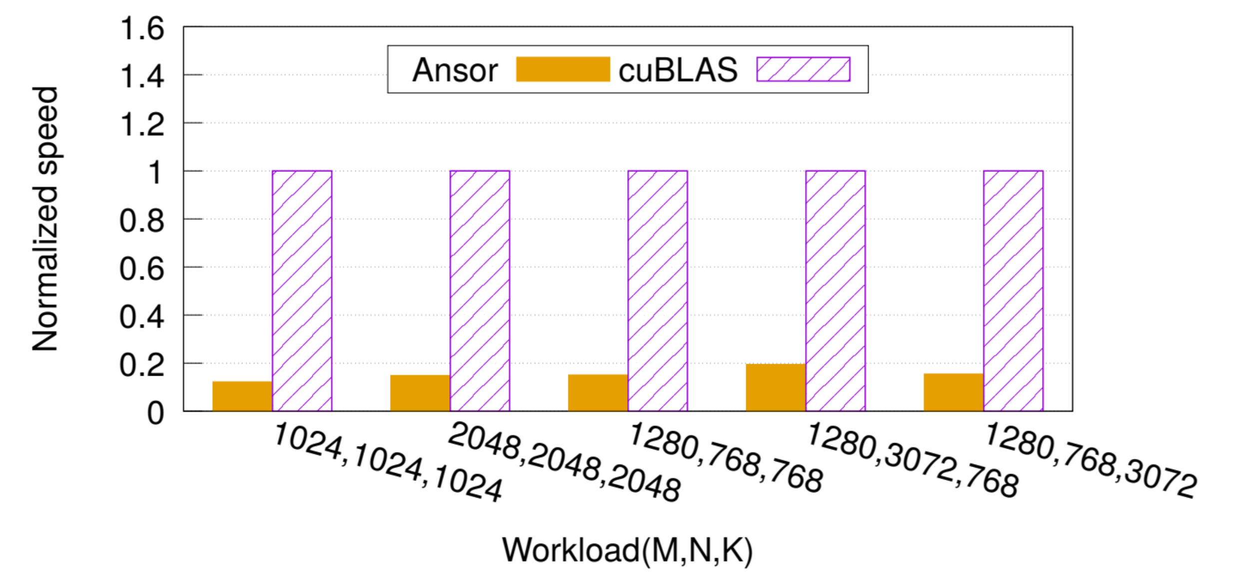


Image source: https://jxing.me/slides/Bolt-mlsys.pdf

Limitations of existing auto-tuners

Neural networks	Tuning time (hours)
ResNet-18	11.8
ResNet-50	15.8
RepVGG-A0	4.4
RepVGG-B0	4.6
VGG-16	19.1
VGG-19	18.8

Long tuning time



Ineffective usage of hardware resources (hardware-agnostic)

UNIVERSITY OF CAMBRIDGE

Key message for this presentation

Bolt auto-tunes end-to-end tensor program optimization in a **hardware-aware manner using templated vendor libraries** to extract the best performance.

UNIVERSITY OF CAMBRIDGE

An emerging trend: Templated libraries (eg: cutlass)

Nvidia's CUDA Templates for Linear Algebra

Optimization technique for fusing two tensor ops (eg: LinearCombination and Relu)

Sm80 means targeting Ampere GPU architecture (eg: A100)

cutlass / examples / 13_two_tensor_op_fusion / **fused_two_convs_f16_sm80_shmem.cu**

Code Blame 236 lines (197 loc) · 8.56 KB

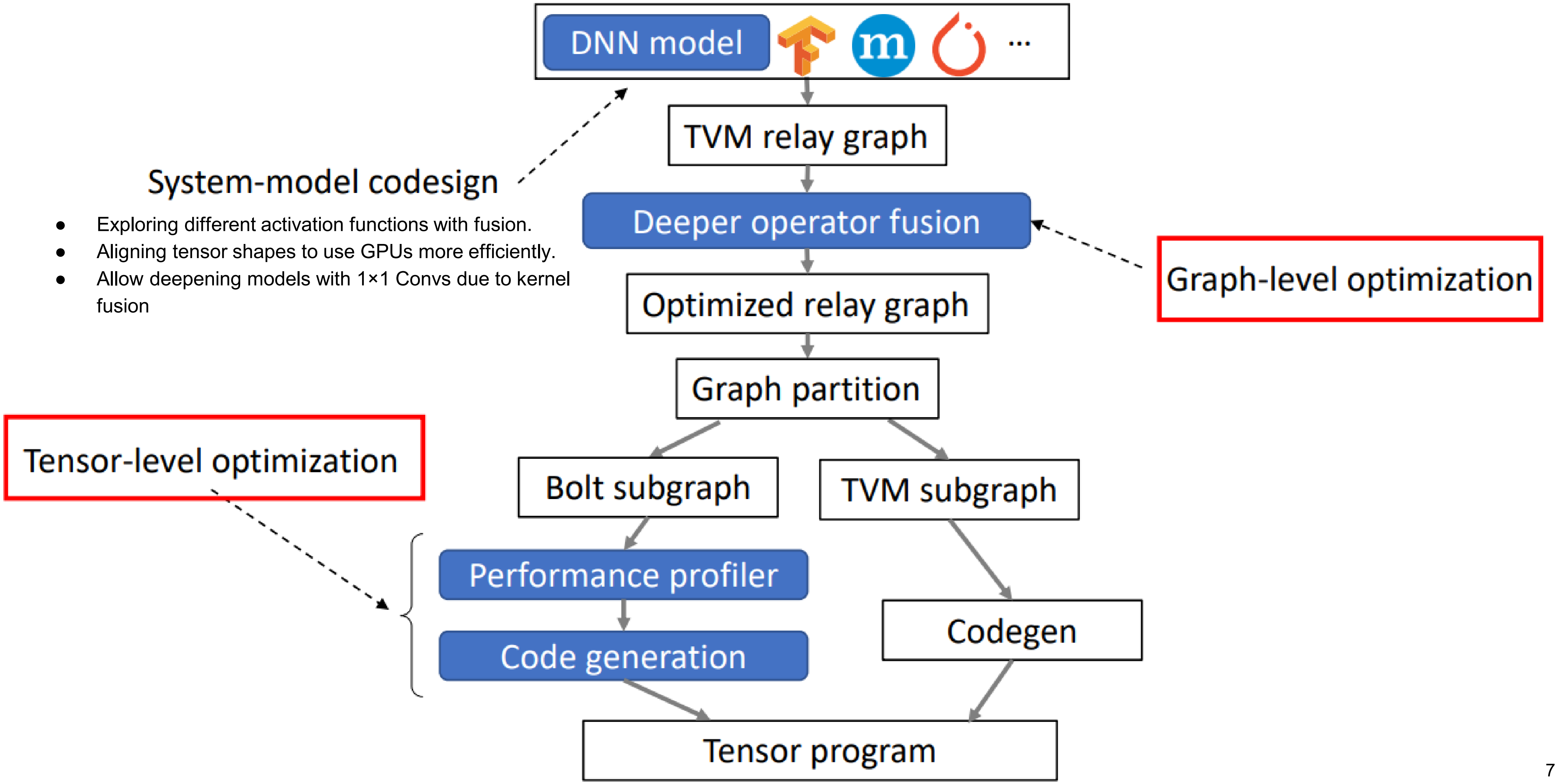
```
  using Conv2dFpropKernel1 = typename cutlass::conv::kernel::DefaultConv2dFprop<
    ElementA, cutlass::layout::TensorNHWC,
    ElementB, cutlass::layout::TensorNHWC,
    ElementC, cutlass::layout::TensorNHWC,
    ElementAccumulator,
    cutlass::arch::OpClassTensorOp,
    cutlass::arch::Sm80,
    ThreadblockShape1,
    WarpShape1,
    InstructionShape,
    cutlass::epilogue::thread::LinearCombinationRelu<
      ElementC,
      128 / cutlass::sizeof_bits<ElementC>::value,
      ElementAccumulator,
      ElementCompute,
      cutlass::epilogue::thread::ScaleType::NoBetaScaling
    >,
    cutlass::gemm::threadblock::GemmIdentityThreadblockSwizzle<1>,
    3,
    cutlass::arch::OpMultiplyAdd,
    cutlass::conv::IteratorAlgorithm::kOptimized
  >::Kernel;
```

This file is a **pre-written, high-performance CUDA C++ kernel template instantiation** designed by NVIDIA to showcase how to **fuse two FP16 convolutions on a modern Ampere GPU using optimized shared memory access.**

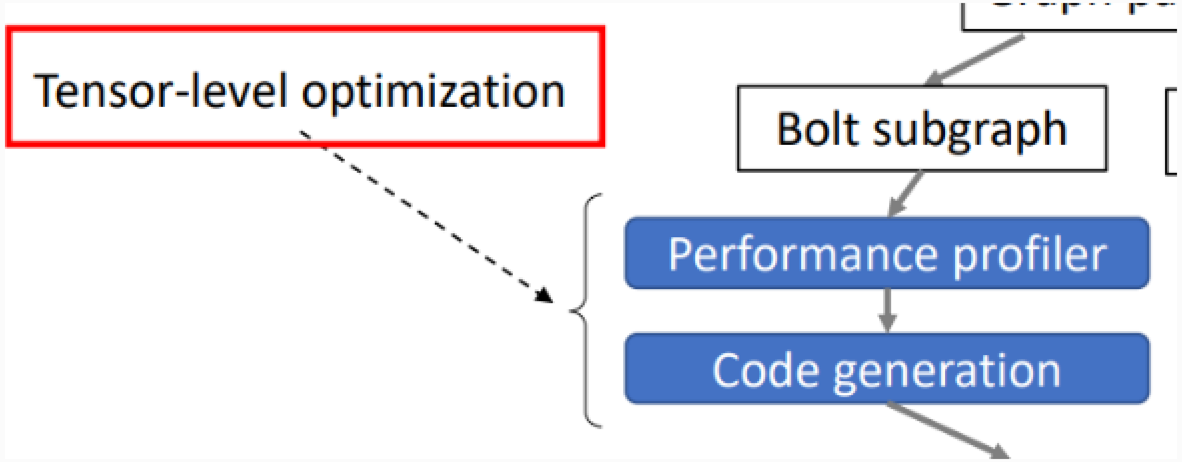
BOLT's auto-tuner aims to **automatically generate this level of configuration**, instead of requiring a human developer to write it.

cutlass::conv::kernel::DefaultConv2DpropKernel……
Template specialization used for modularized composition.

The workflow of BOLT



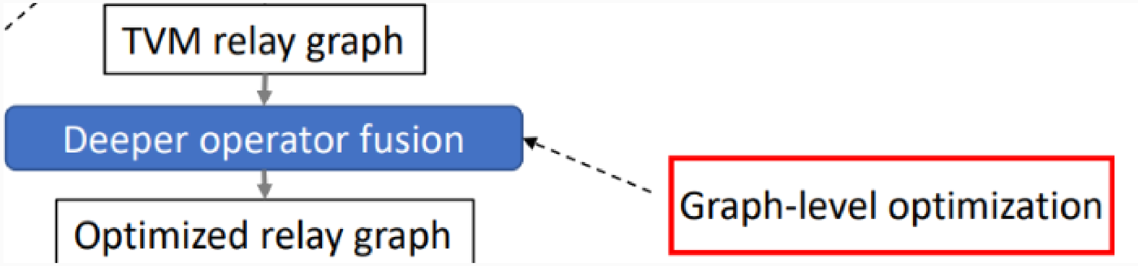
Tensor-level optimization

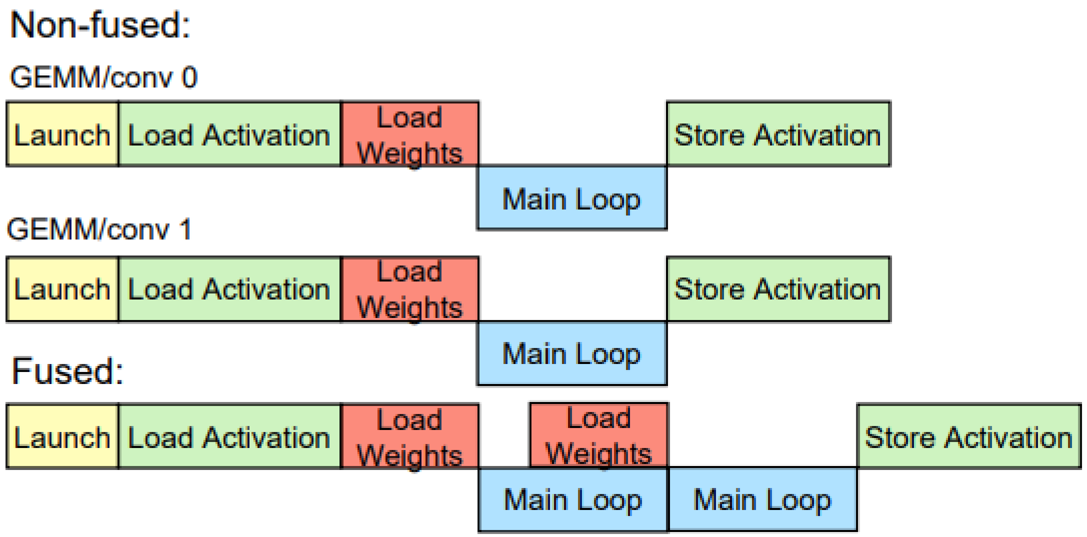


- In Templated code generation
  - Precisely instantiating the parameters to govern tiling sizes and data types creates a high burden.
  - So BOLT has **Light-weight profiler** to search for the best template parameters.
- Besides, BOLT uses these in templated code generation to improve performance
  - layer transformation (Convert NCHW to NHWC - Faster in Conv)
  - kernel padding (to improve on memory coalesced access)

UNIVERSITY OF CAMBRIDGE

Graph level Optimization: Deeper Operator Fusion





(b) The kernel view of persistent kernel fusion

$$D0 = \alpha_0 A0 \cdot W0 + \beta_0 C0, \qquad (1)$$

$$D1 = \alpha_1 D0 \cdot W1 + \beta_1 C1, \qquad (2)$$

Definition of two GEMM fusion

Key ideas:

- Compute the second GEMM without loading its input from the global memory
- Output threadblock of the 1st GEMM in the same threadblock as input for the 2nd GEMM

University of Cambridge

Digression: Reading this paper reminds me of FlashAttention

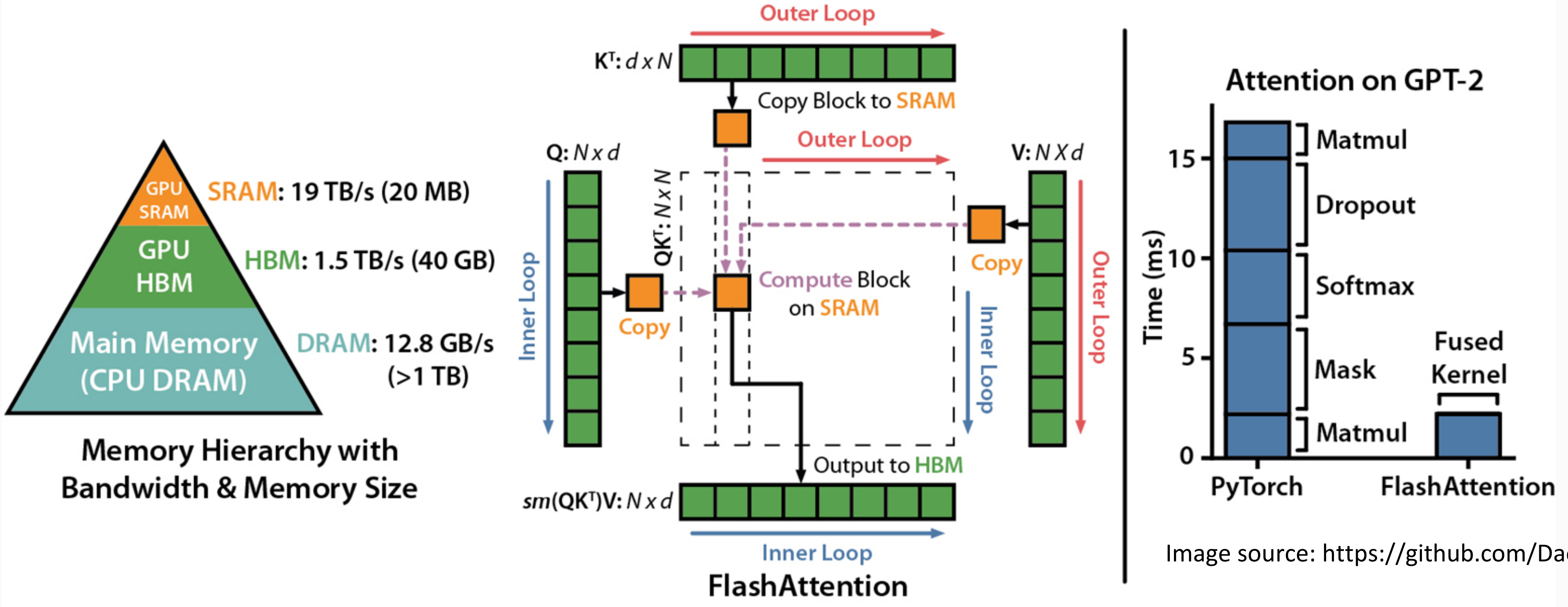
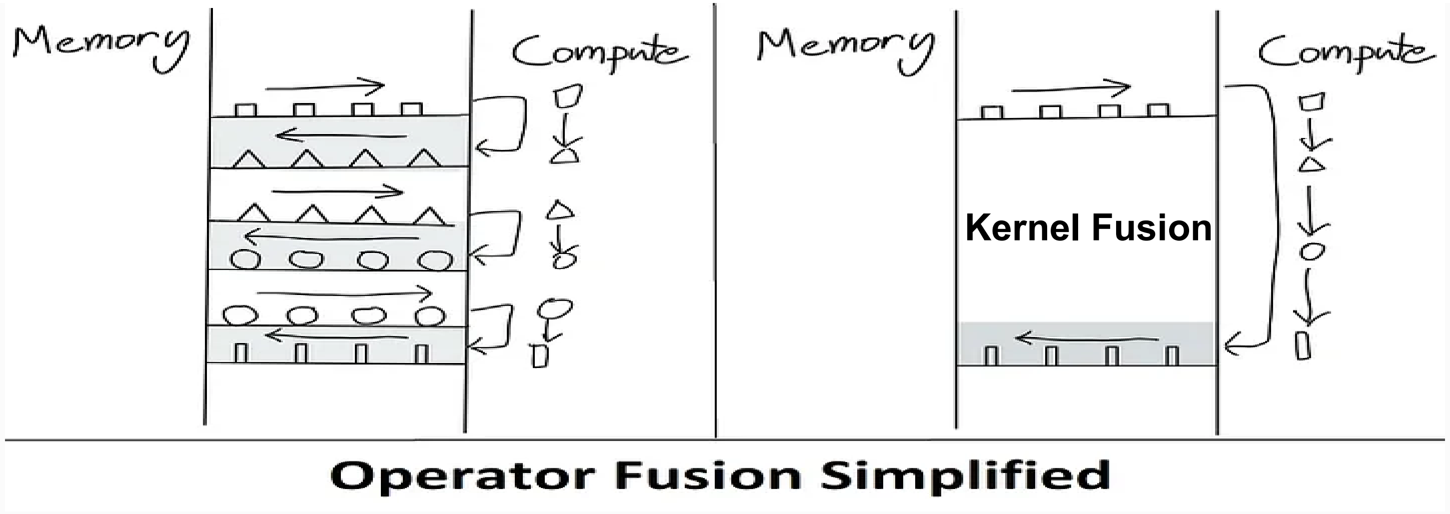
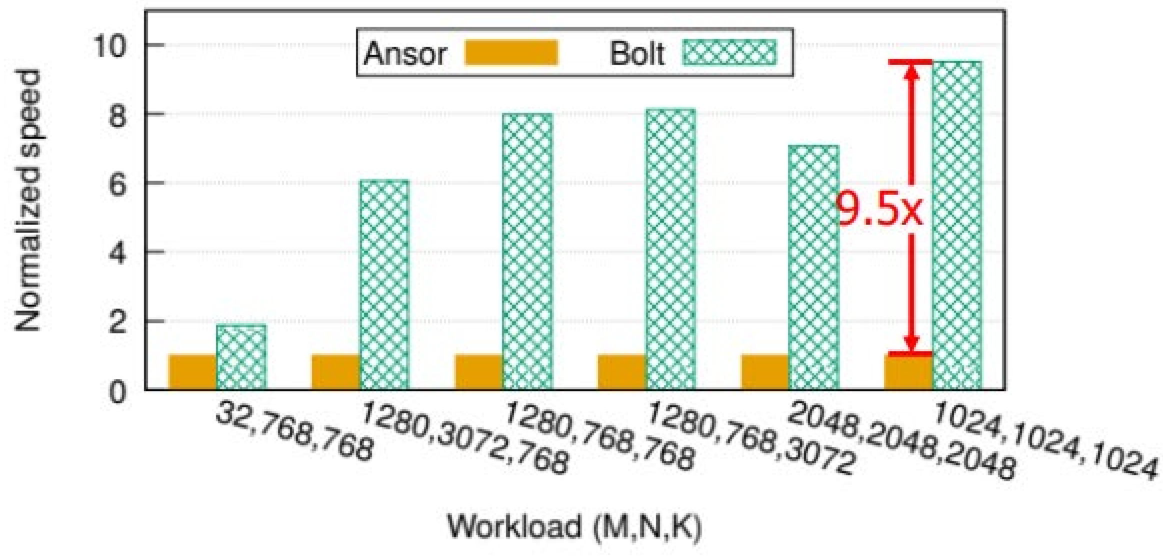


Image source: https://github.com/Dao-AILab/flash-attention

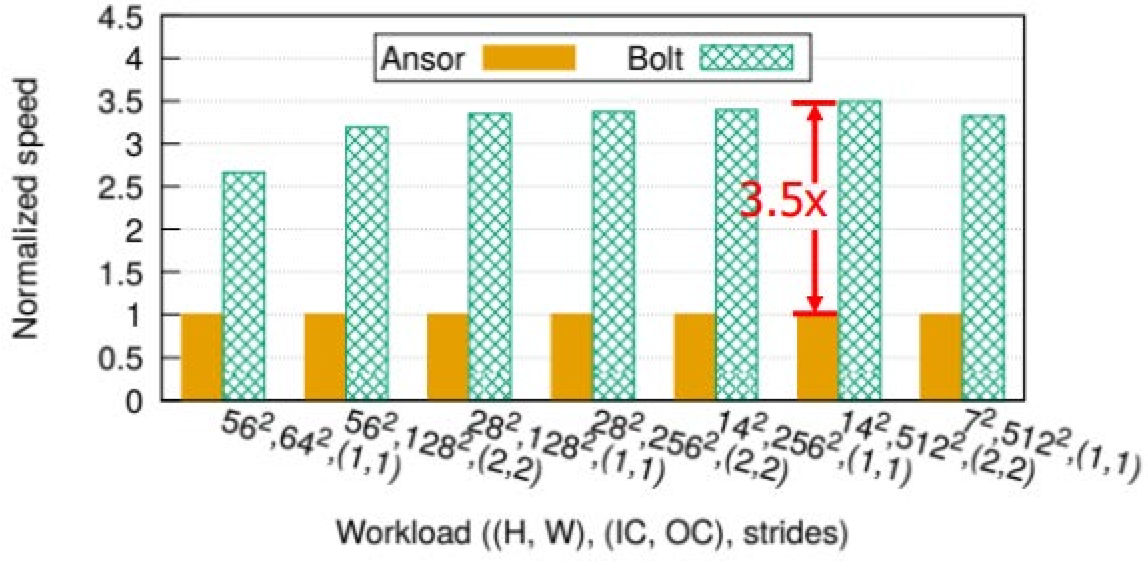


UNIVERSITY OF CAMBRIDGE

Evaluations (GEMM and Conv2D speed)



(a) GEMMs performance.

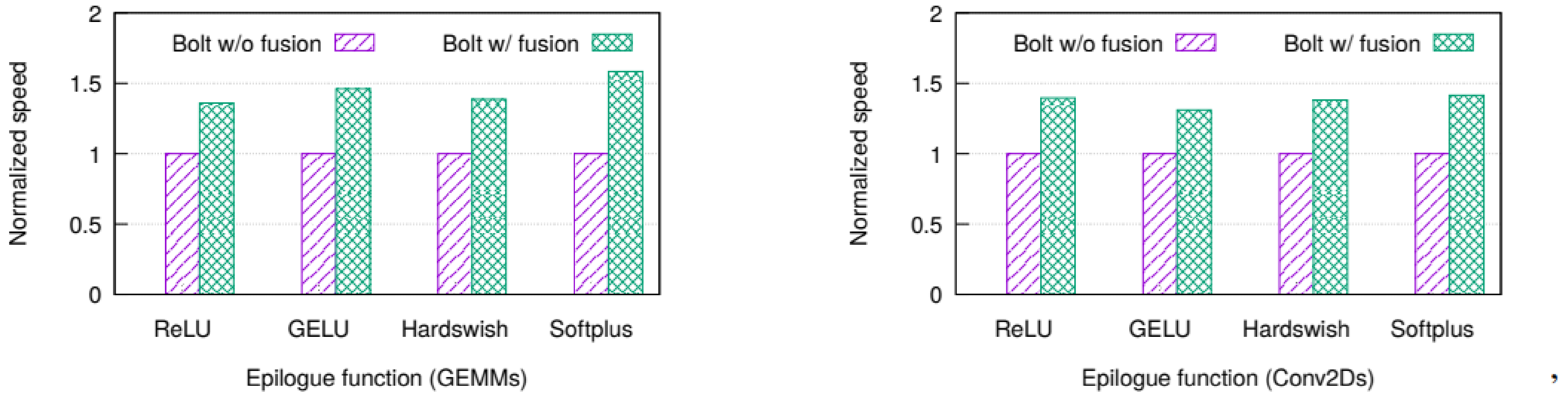


(b) Conv2D performance.

Speedup: 1.9-9.5x for GEMMs, 2.7-3.5x for Conv2Ds

UNIVERSITY OF CAMBRIDGE

Evaluations (Deeper operator fusion performance)

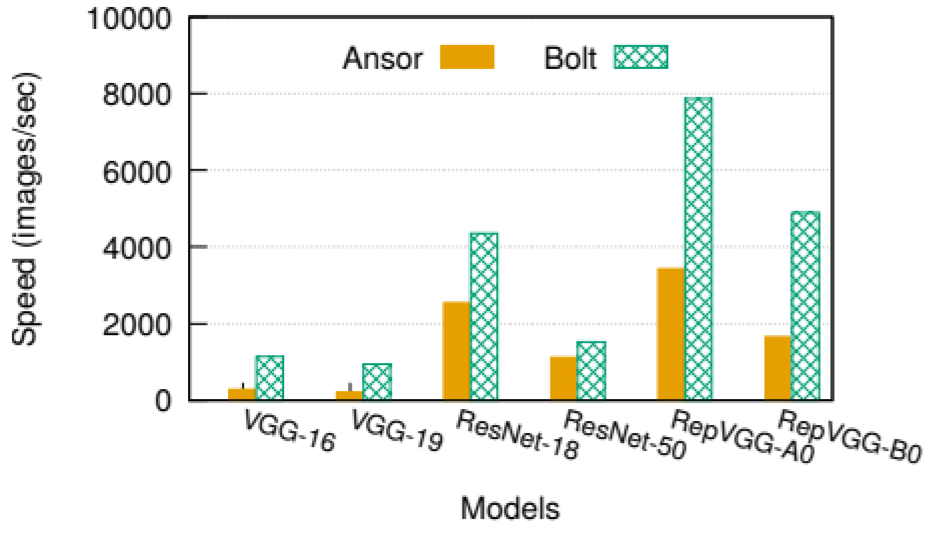


(a) GEMM epilogue fusion. (b) Conv2D epilogue fusion.

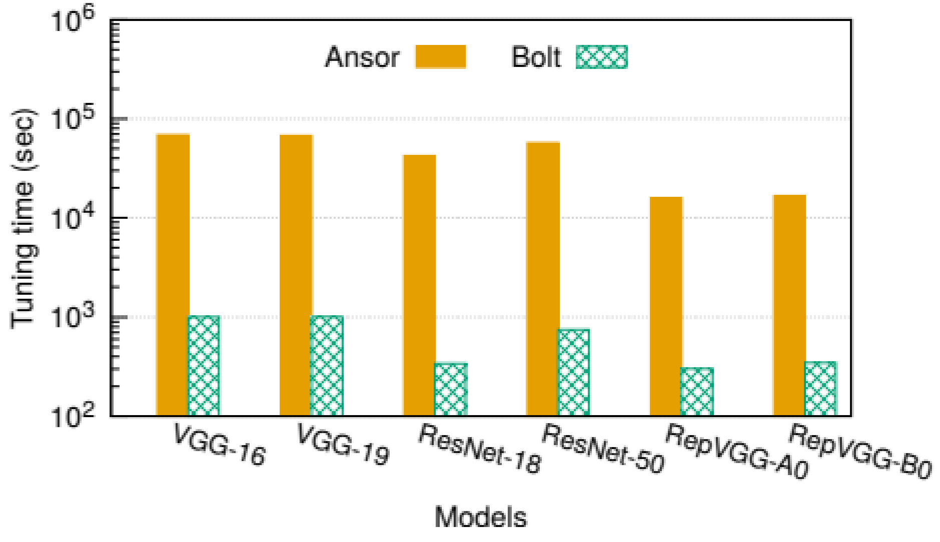
Figure 9. The performance of epilogue fusion on pattern GEMM/Conv2D+BiasAdd+Activation. The workload of the GEMM is M=1280, N=3072, and N=768. The workload of the Conv2d is H=W=56, IC=OC=64, kernel=(3, 3), stride=(1,1), and padding=(1,1).

Speedup: 1.4-1.6x for GEMMs, 1.3-1.4x for Conv2Ds

Evaluations (End-to-end model performance)



(a) Inference speed.



(b) Tunning time.

Bolt is **faster** can finish the tuning **within 20 minutes** for all models while Ansor takes 12 hours on average.

UNIVERSITY OF CAMBRIDGE

Critical Analysis...

Impact and Strength

- Reduce model tuning time from hours to minutes
- Achieve >95% of hardware theoretical performance in FP16 GEMM on Ampere A100
- Enabling Kernel Fusion
- Integrated to tvm (CUTLASS)

Weakness

- Focused on CUTLASS (nvidia), not directly transferable to AMD and Intel GPUs
- Kernel fusion requires intermediate outputs to fit into fast memory (register or shared memory)
    - Only benefits memory bound application but not necessarily for compute bound
- Automated kernel padding in tensor-level optimisation introduces overhead in extra memory allocation
- No multi-objective optimisation that involves energy efficiency

Thank you

Questions?